**3. Operators**

**Operators in JavaScript**:

In JavaScript we use Operators along with our variables and constraint to create expressions. And with this expression we can implement logics and algorithms. In JavaScript we have different type of Operators and they are.

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical operators
5. Bitwise Operators

**1) Arithmetic Operator**:

We use Arithmetic operation for performing calculation. Arithmetic operators take numerical values (either literals or variables) as their operands and return a single numerical value. The basic arithmetic operators are (+, -, \*, /, % and \*\*)

**Example**:

*// Number addition and subtraction*

console.log(2 + 3 - 1); *//4*

*// Number multiplication and division*

console.log((4 \* 3) / 2); *// 12/2 => 6*

*// Number remainder and exponential*

console.log(11 % 3 \*\* 2); *// 11 % 9 => 2*

**2) Assignment Operators**:

An assignment operator assigns a value to its left operand based on the value of its right operand.

**Example**:

var x = 2;

var y = 3;

console.log(x); *//2*

console.log((x = y + 1)); *// 3 + 1 => 4*

console.log((x = x \* y)); *// 4 \* 3 => 12*

**3) Comparison Operators**:

Comparison operators are used in logical statements to determine equality or difference between variables or values.

== equal to

=== equal value and equal type

!= not equal

!== not equal value or not equal type

> greater than

< less than

>= greater than or equal to

<= less than or equal to

**Example**:

let x = 1;

*//relational operator*

console.log(x > 0); *//true*

console.log(x >= 1); *//true*

console.log(x < 1); *//false*

console.log(x <= 0); *//false*

**Equality operators**:

The equality operator converts the operands if they are not of the same type, then applies strict comparison. If both operands are objects, then JavaScript compares internal references which are equal when operands refer to the same object in memory.

console.log(1 == 1); *// true*

console.log("1" == 1); *// true*

console.log(1 == "1"); *// true*

console.log(0 == false); *// true*

console.log(0 == null); *// false*

var object1 = { key: "value" },

object2 = { key: "value" };

console.log(object1 == object2); *//false*

console.log(0 == undefined); *// false*

console.log(null == undefined); *// true*

**Ternary Operator**:

The conditional (ternary) operator is the only JavaScript operator that takes three operands. This operator is frequently used as a shortcut for the if statement.

**Example**:

let points = 100;

let type = points > 100 ? "gold" : "silver";

console.log(type); *//silver*

**4) Logical Operator**:

Logical operators are used to determine the logic between variables or values.

**Example**:

*//Logical AND (&&)*

*//Returns true if both operands ate TRUE*

console.log(true && true); *// true*

*//Logical OR (||)*

*//Returns true if one of the operands is TRUE*

console.log(true || false); *// true*

*//NOT (!)*

*//Always returns the opposit*

console.log(!true); *// false*

**Logical Operators with Non-Booleans**:

If the operator is not boolean true or false it will try to interpreted as "Truthy" or "Falsy"

Falsy(false)

undefined

null

0

false

''

NaN

If we use any of the value in the logical expression we will be treated as "Falsy" which is like as boolean false.

Anything that is not "Falsy" is "Truthy"

**Short-circuiting**:

Two important aspects of logical operators in JavaScript is that they evaluate from left to right, and they short-circuit

true || \*\*\*\*

// tru

**5) Bitwise Operator**:

Bitwise operators treat their operands as a sequence of 32 bits (zeroes and ones), rather than as decimal, hexadecimal, or octal numbers. For example, the decimal number nine has a binary representation of 1001. Bitwise operators perform their operations on such binary representations, but they return standard JavaScript numerical values.

**Example**:

*//BitWise OR*

*// 1 = 00000001*

*// 2 = 00000010*

*// 3 = 00000011*

console.log(1 | 2); *//3*

*//BitWise AND*

*// 1 = 00000001*

*// 2 = 00000010*

*// 3 = 00000000*

console.log(1 & 2); *//0*

**Example**:

*//user has Read, Write, Execute permission*

*//Read permission -> 00000100*

*//Write permission -> 00000010*

*//Execute permission -> 00000001*

const readPermission = 4;

const writePermission = 2;

const executePermission = 1;

let myPermission = 0;

myPermission = myPermission | readPermission | writePermission;

console.log(myPermission); *//6*

let message = myPermission & readPermission ? "yes" : "no";

console.log(message); *//yes*

**Swapping Value**:

let a = "read";

let b = "blue";

let c = a;

a = b;

b = c;

console.log(a); *//blue*

console.log(b); *//read*

3. Operators